Camera Based Object Tracking

Mentor: Prof. Bakul Gohel

Jainil Parmar - 201901420
Kishan Gajnotar - 201901116

Object Tracking

- To track an object, we need its location.
- In this case, there will be two types of locations.
  - 3D coordinates (actual world coordinates)
  - Coordinates of the 2D projection from the camera.
- These 2D coordinates depend on some camera parameters:
  - Intrinsic Parameters
  - Extrinsic Parameters
- Intrinsic Parameters are the parameters which belong to the camera itself like focal length, optical center, distortion, etc.
- Extrinsic Parameters are the parameters which refers to the orientation of the camera with respect to some world coordinate system likely rotation and translation.
- And to find these parameters we do camera calibration.

$$\mathbf{P} = \overbrace{\mathbf{K}}^{\text{Intrinsic Matrix}} \times \overbrace{[\mathbf{R} \mid \mathbf{t}]}^{\text{Extrinsic Matrix}}$$

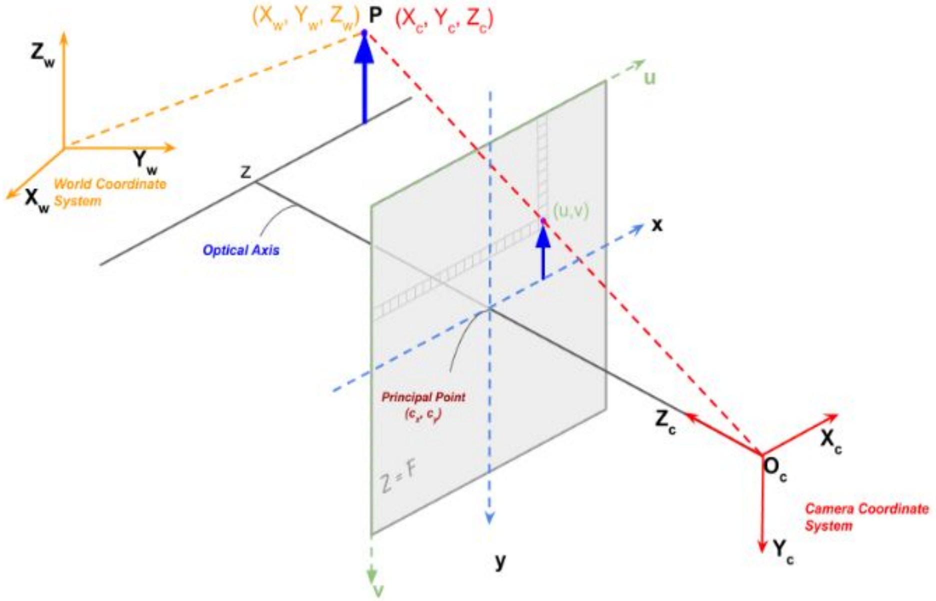
- Here, P is the projection matrix.

Camera Calibration

If we take (u,v) as the 2D coordinates of the projection, (X_W, Y_W , Z_W) as the world coordinates of the object and (X_C, Y_C , Z_C) as the coordinates of the object w.r.t the camera, we can define the 2D coordinates as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [\mathbf{R}|\mathbf{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Here, **R** is the rotational matrix, and **t** is the translational matrix.



$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

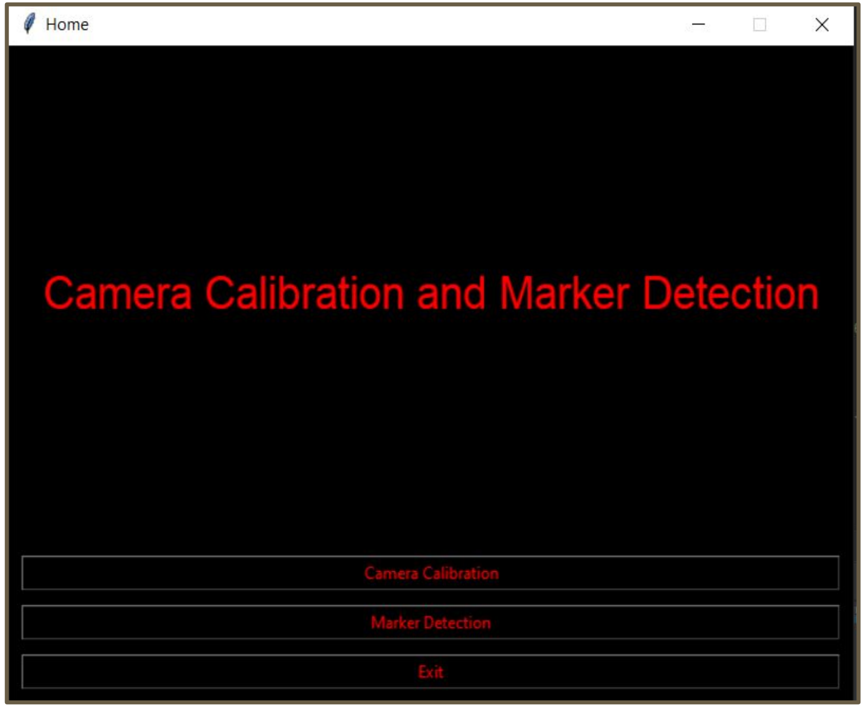
$$u = \frac{u'}{w'} \qquad v = \frac{v'}{w'} \qquad \mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

What we did

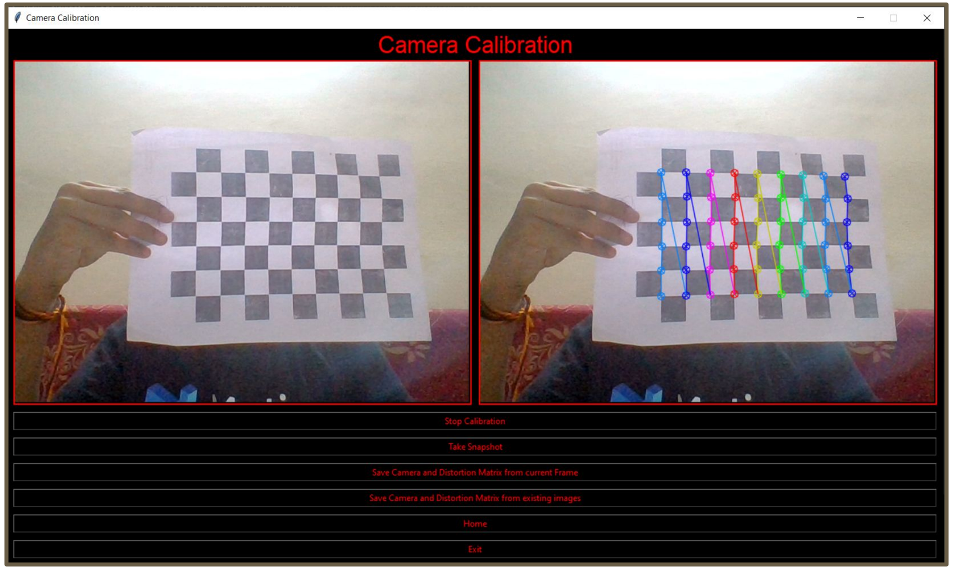
- We have used the OpenCV python library (a real-time computer vision library) for the purpose of Camera calibration and object detection.
- For camera calibration, we used the checkerboard calibration method.
- There are built-in functions in OpenCV for the camera calibration.
- After we get the parameters, we used them to detect two ArUco markers and give real-time 3D distance between them.
- As, the functions to detect ArUco markers give 2D coordinates, we used the camera Parameters to get the 3D world coordinates and find real-time distance between two markers.
- After doing these things, we integrated these functions into a Python Tkinter GUI.
- This app will be used as a utility for a different project in the future.
- The video link of the working GUI: SRI_demo.mkv

GUI

- Home Page.
- From this page we can navigate to both the functions for calibration and detection.



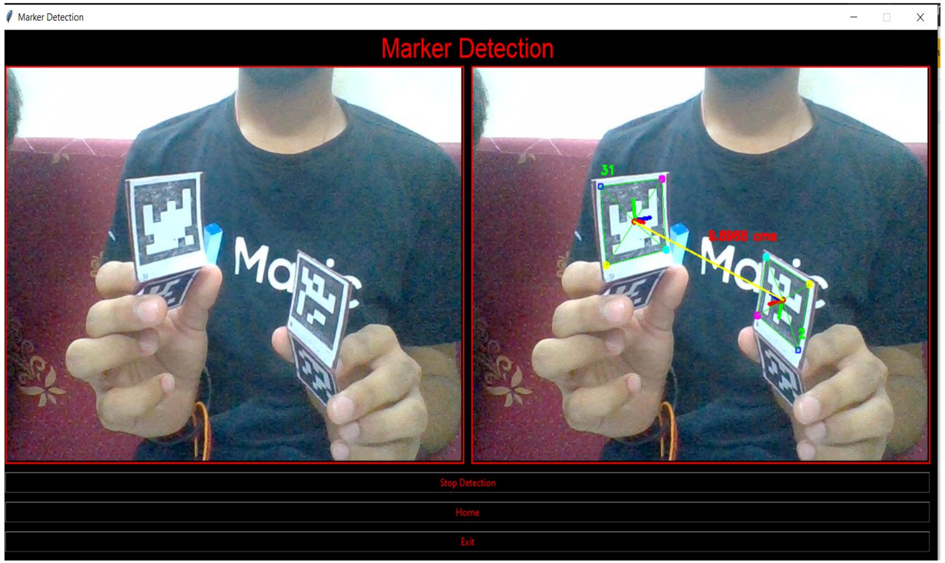
GUI


Camera Calibration
Camera Calibration
Stop Calibration
Take Snapshot
Save Camera and Distortion Matrix from current Frame
Save Camera and Distortion Matrix from existing images
Home
Exit

- This screen shows the checkerboard calibration technique.
- This method detects the corners of the squares in the checkerboard of given size.
- After detecting the checkerboard pattern, the camera matrices can be determined.
- There are functions to save snapshots, save matrices from already existing photos and save the matrices from current frame.

GUI

- This screen shows the marker detection and real-time distance.
- After loading the matrices, and setting up the ArUco marker IDs, we can detect the ArUco markers and the distance.



What we Learnt

- Camera Calibration Methods.
- Object Detection Methods.
- Object Measurement Methods.
- Real-time 3D plotting.
- Python Tkinter, OpenCV, ArUco, Pillow

Future Goals

- Real-time marker corners on a 3D plot.
- Pivoting of the points and using them as reference points.
- The Ultimate goal is to use the product for the tracking purpose in a Knee Replacement Surgery.

References

- Camera Calibration using OpenCV
- Geometry of Image Formation
- Aruco markers with openCv, get the 3d corner coordinates? - Stack Overflow
- OpenCV: Camera Calibration
- Detection of ArUco Boards

Thank You